# Laboratorio 2 – Creación y manipulación de Clases en C++

# GUIÓN DEL LABORATORIO

## 1.- Objetivos del laboratorio

- Comprender los conceptos básicos relativos a Clases y Objetos.
- Identificar los componentes básicos de una clase en C++, incluyendo elementos públicos y privados.
- Construcción de proyectos de programación compuestos por varios archivos de código.
- Diseñar y escribir una clase simple.

## 2.- Antes de asistir al laboratorio

Antes de asistir al laboratorio debes realizar las siguientes tareas:

1. Leer alguna de las lecturas recomendadas y los apuntes de clase sobre clases en C++.
2. Leer el documento con instrucciones para el manejo de proyectos en *DevC++*.
3. Revisar el guión del laboratorio (este documento).
4. Analizar el código que se proporciona y que está en la página web de la asignatura.
5. Contestar a la cuestiones 1 a 5 de la Actividad 0 de la hoja de Prerrequisitos. Recuerda que es preciso entregar las respuestas a esta actividad antes de la fecha prevista. En caso contrario, no podrás acceder al laboratorio ni entregar de ninguna manera la solución a esta práctica.

## 3.- Actividades a realizar

Actividad 1: Comprender los componentes de una clase C++.

1. Crear un nuevo directorio llamado `Lab2` en el escritorio.
2. Copiar los archivos "`carta01.h`", "`carta01.cpp`", "`testCarta01.cpp`" de la página web de la asignatura en el directorio `Lab2`.
3. Abrir *DevC++*
4. Crear un nuevo proyecto de tipo "Empty Project" con el nombre "`carta_act1`" en el directorio `Lab2`.
5. Añadir los archivos "`carta01.h`", "`carta01.cpp`", "`testCarta01.cpp`" al proyecto "`carta_act1`"
6. Reconstruir todo el proyecto y contestar a las cuestiones 6 y 7 de tu Hoja de trabajo.
7. Guardar los cambios realizados y cerrar el proyecto (Archivo >> Cerrar proyecto).

Actividad 2: Añadir operaciones a una clase ya existente y modificar operaciones.

1. Guardar los archivos "`carta01.h`" y "`carta01.cpp`" con el nombre "`carta02.h`" y "`carta02.cpp`", respectivamente.
2. Crear un nuevo proyecto de tipo "Empty Project" con el nombre "`carta_act2`" en el directorio `Lab2`.
3. Añadir los archivos "`carta02.h`", "`carta02.cpp`", "`testCarta02.cpp`" al proyecto "`carta_act2`".
4. Modificar las operaciones `Carta::PonerPalo` y `Carta::PonerNumero` para que no sea posible asignar valores incorrectos al palo y al número de una carta. Los únicos valores válidos para el palo deben de ser: 'O', 'C', 'E' y 'B' (oros, copas, espadas y bastos). Los

números válidos para una carta son los números enteros comprendidos entre 1 y 12 (incluidos). Cuando se detecte algún error en los argumentos se deben mostrar los siguientes mensajes por pantalla (según el caso):

```
"Carta::PonerPalo, Argumento incorrecto"
"Carta::PonerNumero, Argumento incorrecto"
```

5. Implementa en la clase carta dos nuevas operaciones `Carta::Palo()` y `Carta::Numero()`, que devuelvan el número y el palo de la carta, de manera que el programa "`testCarta02.cpp`" funcione correctamente.
6. Reconstruir todo el proyecto y contestar a la cuestión 8 de tu Hoja de trabajo.
7. Guardar los cambios realizados y cerrar el proyecto.

Actividad 3: Construir una nueva clase que cumpla un conjunto de especificaciones.

1. Crear un nuevo proyecto de tipo "Empty Project" con el nombre "`carta_act3`" en el directorio `Lab2`.
2. Añadir los archivos "`carta02.h`", "`carta02.cpp`", "`testBaraja.cpp`" al proyecto "`carta_act3`".
3. Crear y añadir al proyecto los archivos "`baraja.h`" y "`baraja.cpp`". Estos archivos, inicialmente vacíos, contendrán el interfaz y la implementación de la clase `Baraja`, que debe cumplir los requisitos que se enumeran a continuación:
   a. Una baraja es un conjunto de 48 cartas, agrupadas en cuatro palos diferentes (oros, copas, espadas y bastos) y numeradas de 1 a 12 dentro de cada uno de los palos.
   b. Sólo hay cuatro operaciones permitidas sobre la baraja:
      i. `void IniciarCartas ()`
         La operación asigna valores a las 48 cartas (número y palo) y las agrupa de manera que, las 12 primeras cartas serán las correspondientes a los "oros", numeradas en orden ascendente. A continuación irán todas las cartas de "copas", con el mismo criterio de orden. Después las 12 cartas de "espadas" y, finalmente, las 12 cartas de "bastos"
      ii. `void Barajar ()`
         La operación mezcla todas las cartas de la baraja, de manera que el orden de las 48 cartas sea aleatorio. Se recomienda para ello, seguir la estructura de la función `GenerarAleatorio` que se utilizó en la primera práctica y adaptarla a esta situación.
      iii. `void VerBaraja ()`
         La operación debe mostrar por pantalla las 48 cartas de la baraja en el orden en que están almacenadas.
      iv. `void OrdenarCartas ()`
         La operación debe ordenar todas las cartas de la baraja. Para ello, se utilizará el algoritmo de ordenación por *Selección*, adaptado convenientemente a este caso (se ordenan cartas y no enteros), y se utilizará como criterio para la ordenación el establecido por la operación `Carta::MenorCarta`.
4. Verificar el correcto funcionamiento de la clase `Baraja` mediante la ejecución del programa "`testBaraja.cpp`" que forma parte del proyecto.
5. Contesta a las preguntas 9, 10 y 11 de tu Hoja de trabajo.

**Los archivos con los programas correspondientes a las actividades se deberán entregar, a través del aula virtual, al finalizar la sesión de prácticas, junto con la Hoja de Trabajo del estudiante. Los archivos C++ que se deben entregar son:**

- `testCarta01.cpp`
- `carta02.h`
- `carta02.cpp`
- `baraja.h`
- `baraja.cpp`

## 4.- Después de asistir al laboratorio

Una vez finalizada la práctica, contesta a las cuestiones correspondientes a la Actividad 4 que aparecen en el Cuestionario Posterior al laboratorio.

## 5.- Lecturas recomendadas

- Capítulos 6 (pp. 305-352) y 8 (pp.479-505) de "*Resolución de problemas con C++ (2nd. Ed.)*", W. Savitch, PrenticeHall (2000).
- Capítulo 13 de "*Programación en C++. Algoritmos, estructuras de datos y objetos (2ª ed.)*", L. Joyanes. McGrawHill (2006)